RASPBERRY PI, TNC-PI en APRX

Installatie document – Updated 3-09-2016

VOORBEREIDING 2

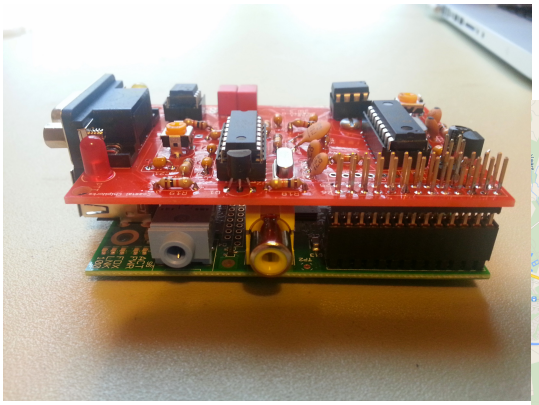
AANPASSEN STANDAARD CONFIG VOOR GEBRUIK I2C 2

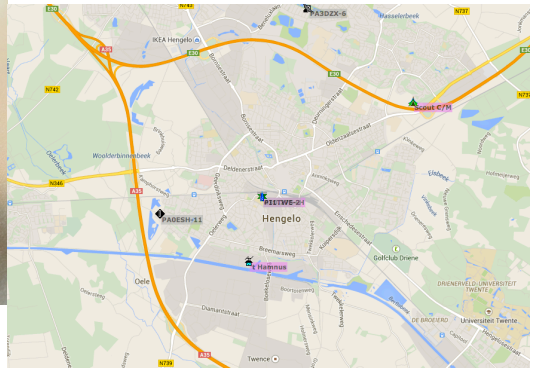
TESTEN I2C 4

GEBRUIK MET KISS PROGRAMMA EN COM POORT 6

INSTALLEREN VAN APRX VIA GITHUB 6

AUTOSTART APRX OP RASPBERRY 8





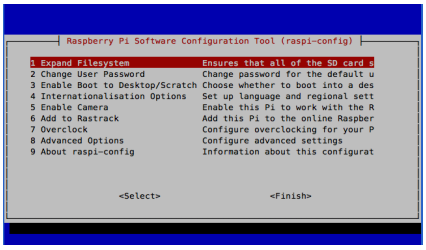
Let op: Er zijn wat zaken veranderd t.o.v. van de vorige versies van Raspbian wheezy en i2c.

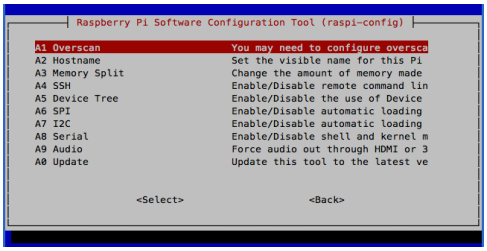
Voorbereiding

Dit artikel gaat er vanuit dat u zelf een nieuwe verse image op de sd-kaart hebt gezet en wel een wheezy installatie, met de naam ***2015-05-05-raspbian-wheezy.img***

- Versie: **Mei *2015***
- Release date: ***2015-05-05***
- ***http://www.raspberrypi.org/downloads/***

Connect met de Raspberry PI via ssh als gebruiker pi (eerste keer is wachtwoord raspberry).
Doe eerst de basis zaken via het commando **sudo raspi-config** en pas vooral de **advanced options** aan, zodat i2c opstart !!!!!





Kies voor **reboot** aan het einde, log opnieuw in en voer de upgrade naar de laatste stand uit met de volgende commando's:

sudo apt-get update
sudo apt-get dist-upgrade
sudo rpi-update

Reboot weer (sudo reboot)

Aanpassen standaard configuratie RPI voor gebruik i2c

Nadat we ingelogd zijn op de Raspberry schakelen we over naar de superuser su

sudo su

Allereerst passen we de file **cmdline.txt** aan, omdat we geen seriële communicatie gaan gebruiken maar i2c en dit soms in de weg staat.:

cd /boot

nano cmdline.txt
Gooi alles weg wat er staat, (Ctrl-k op de regel in nano) en plaats de tekstregel hieronder in de plaats:

dwc_otg.lpm_enable=0 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline rootwait

Opmerking: In de nieuwe versie van wheezy is er geen **/etc/modprobe.d/raspi-blacklist.conf** file meer. In de oude wheezy moest je het volgende doen.

In de **/etc/modprobe.d/raspi-blacklist.conf** bestand, verwijderen we de regel: ***blacklist i2c-bcm2708 door er een hekje voor te zetten***

```
En dan ziet het er zo uit…..
blacklist spi-bcm2708
#blacklist i2c-bcm2708
blacklist snd-soc-pcm512x
blacklist snd-soc-wm8804
```

Wel moeten we de i2c-dev en i2c-bcm2708 modules toevoegen in /etc/modules:

nano /etc/modules, en dan voegen we twee regels toe: **I2C-dev en i2c-bcm2708**
En dan ziet dat er zo uit:

```
# /etc/modules: kernel modules to load at boot time.
#
# This file contains the names of kernel modules that should be loaded
# at boot time, one per line. Lines beginning with "#" are ignored.
# Parameters can be specified after the module name.

snd-bcm2835
i2c-dev
i2c-bcm2708
```

Ook moet we (helaas) nog in de config.txt file aangeven dat i2c geladen moet worden.
Pas ***/boot/config.txt*** als volgt aan - het gaat om de extra regel **dtparam=i2c1=on**
aan het einde:

```
core_freq=250
sdram_freq=450
over_voltage=2
dtparam=spi=off
dtparam=i2c_arm=on

dtparam=i2c1=on
```

Installeer nu de ax25 zaken en i2c support met het commando
apt-get install ax25-tools ax25-apps i2c-tools

Vervolgens de de file ***/etc/ax25/axports*** aanpassen en vooral de lege regels er uit laten, dus die van mij ziet er zo uit:

```
# /etc/ax25/axports
# The format of this file is:
# name callsign     speed paclen window   description
#
vhf1    PA0ESH-10    19200    236      2     144.800 MHz (1200  bps Europe)
uhf1    PA0ESH-11    19200    236      2     432.500 Mhz (1200  bps Europe)
```

TNC-Pi setup voor de I2C bus.

De TNC-Pi NVRAM is geconfigureerd voor seriële bediening als je 'm voor het eerst aansluit. De TNC-Pi kan echter ook worden geconfigureerd om te communiceren met de Raspberry Pi via het I2C protocol, en dat gaan we maar eens doen.

Om de TNC-Pi voor I2C te configureren, heb je de programma's **pi_tncsetparams** en **pi_tncgetparams**

nodig. Het **getparams** programma leest de parameters van de TNC-Pi, terwijl het **setparams** programma je in staat stelt om de parameters in te stellen.
Deze programma's zijn beschikbaar op: https://dl.dropboxusercontent.com/u/31910649/pitncgetset.zip

Download de Param software, extract en plaats de **pitnc_getparams** en **pitnc_setparams** en **i2ckiss** bestanden in de /usr/local/sbin/ directory.
cd /usr/local/sbin/
wget www.tnc-x.com/i2ckiss.zip
unzip i2ckiss.zip
wget https://dl.dropboxusercontent.com/u/31910649/pitncgetset.zip
unzip pitncgetset.zip
rm –rf *.zip

en maak ze uitvoerbaar met

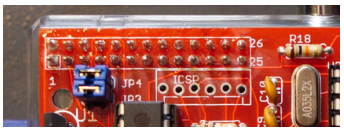
chmod u+x pitnc_setparams
chmod u+x pitnc_getparams
chmod u+x i2ckiss
reboot

En log weer in als root. (ssh root@xx.xx.xx.xx)

Als je de niet-officiële ax25 bron gebruiken moet je een link toe te voegen aan de RT Library routines.
ln -s /usr/local/lib/libax25.so.1 /usr/lib/libax25.so.0

Configureer de TNC-Pi boards met één tegelijk.

Verwijder de eventueel andere TNC-PI boards vanaf de top van de Raspberry of verwijder JP 7 (power) van de andere TNC's.
Jumpers JP 3 en 4 moeten op de TNC worden gezet om te kunnen configureren. Tijdens normaal werk, moeten ze er vanaf.
.
Tussendoor : Als default heeft de gebruiker PI geen toegang tot de i2c bus.
Je kunt natuurlijk alles als root draaien , maar het lijkt beter om ook gebruiker pi (of wat je zelf hebt aangemaakt) toegang tot de i2c bus te geven
usermod -a -G i2c pi



Nogmaals, check Jumpers **JP 3 & 4** of deze zijn geïnstalleerd op de TNC-PI en in de juiste richting. Parrallel aan de GPIO pinnen.

Testen i2c

Laten we eerst eens kijken of we wat zien op de i2c bus:
Doe dit vanuit de directory **/usr/local/sbin !!**

Het commando : **i2cdetect –y 1** levert bij mij het volgende op**:**

```
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: 10 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
```

```
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
root@tnc-pi:/usr/local/sbin#
```

Ik zie dus dat de TNC-PI op adres 10 staat: Dat is in HEX 16.
En omdat ik en nieuwere Raspberry heb is het **bus 1** bij mij.
M.a.w. de TNC-PI heeft **bus 1** en **device 16** (in HEX)

http://www.binaryhexconverter.com/decimal-to-hex-converter

Lees nu eerst de KISS Parameters uit met het commando
./pitnc_getparams (b) (d)
Oftewel

```
01 TXDelay - Zero means use ADC  40
02 Persistance                   63
03 Slottime (in 10 mS)           20
04 TXTail                        30
05 Full Duplex - Not used         0
06 Our Channel (Hex)             0d
07 I2C Address (0 = async) Hex   10
   ADC Value                     31
8 0 28 3f 14 1e 0 d 10 1f 17 c0 sum 0
root@tnc-pi:/usr/local/sbin#
```

Als je parameters wilt veranderen, dan doe je dat met **/usr/local/sbin/*pitnc_setparams (b) (d) (parm) (value)***

Voorbeeld: ik wil de TX delay naar 50 zetten

./pitnc_setparams 1 16 1 50

```
root@tnc-pi:/usr/local/sbin# /usr/local/sbin/pitnc_setparams  1 16 1 50
8 0 32 3f 14 1e 0 d 10 1f d c0 sum 0

01 TXDelay - Zero means use ADC  50
02 Persistance                   63
03 Slottime (in 10 mS)           20
04 TXTail                        30
05 Full Duplex - Not used         0
06 Our Channel (Hex)             0d
07 I2C Address (0 = async) Hex   10
   ADC Value                     31
root@tnc-pi:/usr/local/sbin#
```

Verwijder nu de jumpers 3 & 4 van de geconfigureerde TNC-PI en reboot.

Uitvoeren van applicaties met TNC-Pi in de I2C-modus

De TNC-PI kan worden gebruikt met toepassingen die de Linux AX.25 stack gebruiken, of toepassingen die verwachten dat er een KISS TNC te zien op een seriële poort.
Het i2ckiss programma dat I2C converteert naar een standaard KISS-protocol via een virtuele seriële (PTY) poort, is te downloaden van http://www.tnc-x.com/i2ckiss.zip

Gebruik met ax25 stack

Om een AX.25 socket te maken, gebruiken we i2ckiss in plaats van een kissattach te creëren.

Voer een proc van i2ckiss uit voor iedere TNC-PI met het commando :
i2ckiss <bus> <device> <axport> 44.142.1.1

De parameters die we gebruiken bij het i2ckiss commando zijn :
I2C bus, I2C-apparaat, gevolgd door de naam van de poort (vanuit axports, in mijn geval dus vhf1 en uhf1) en een ip-adres.

Het adres zal waarschijnlijk niet worden gebruikt voor het routeren, zodat het AmpR Net Test adres - 44.128.1.1 veilig gebruikt kan worden.
Voorbeeld:
i2ckiss 1 16 vhf1 44.128.1.1 **(**voor de 2 meter versie).
Dit zal een PTY-pair te maken en uit voeren als kissattach met behulp van de 3e (vhf1)en 4e (44.128.1.1)parameters.

Het is altijd goed om eventuele instantiaties van i2ckiss uit te schakelen met het commando:
killall i2ckiss.
.

Dit is bij mij het resultaat

```
root@tnc-pi:/usr/local/sbin# killall i2ckiss
root@tnc-pi:/usr/local/sbin# killall i2ckiss        (voor de zekerheid)
i2ckiss: no process found
root@tnc-pi:/usr/local/sbin# i2ckiss 1 16 vhf1 44.128.1.1
i2ckiss Version 0.1.0
slave device is: /dev/pts/5
Resetting TNC...
AX.25 port vhf1 bound to device ax0
root@tnc-pi:/usr/local/sbin#
```

Ik kan nu de ax25 port vhf1 gebruiken in programma's zoals aprx etc
Heb je twee TNC-Pi's, dan doe je dezelfde stappen nog een keer, maar dan met de eerste TNC-PI ontkoppeld.
En uiteraard gebruik je dan op het einde (en ook later tijdens het gewone werken) een ietwat andere proc:
Bv:

Voor TNC-Pi 1: **`i2ckiss 1 16 vhf1 44.128.1.1`**
Voor TNC-Pi 1: **`i2ckiss 1 32 vhf2 44.128.1.1`**

Dit moet je aan het einde dus ook zo in het opstart scriptje stoppen. Het mooie van aprx is, dat je het ook als cross band digi kunt inzetten….

Gebruik met KISS programma en com poort

Als het een KISS programma betreft, zoals bv Xastir, dan voeren we (eenmalig) we het volgende uit:

./i2ckiss 1 16 symlink com1

i2ckiss creëert dan een pty paar, en daarna een symlink naar com1. In de applicatie dien je dan com1 te gebruiken als KISS com poort.

Opmerking: als je hiervoor kiest – of voor een mix, dan moet dat ook allemaal via een opstart scriptje aangemaakt worden, bij het booten.

Installeren van aprx via github

Om aprx te installeren en zeker te zijn van de laatste versie is het aan te bevelen om dit steeds zelf te compileren uit de github repository.
Er is op dit moment een fork aktief van Phil die prima werkt. Matti is eigenlijk een beetje gestopt met de ontwikkeling. Zie https://github.com/PhirePhly/aprx

Allereerst installeren we eenmalig de ontwikkelomgeving.

- git-core - Git control systeem, voor verbindingen met de repository
- scons - Build Systeem
- libsdl1.2-dev - SDL Libaries
- libbz2-dev - BZip2 Libaries - 64bit systems use lib64bz2-dev
- liblua5.1-0-dev - Lua libaries
- build-essential - Compiling tools
- libfftw3-dev - noodzakelijk.

Installeer alles met het volgende commando:

sudo apt-get install scons git-core build-essential libsdl1.2-dev libbz2-dev liblua5.1.0-dev fftw3-dev screen

We halen nu de source code van aprx op met

sudo git clone https://github.com/PhirePhly/aprx

ga naar de gecreëerde directory aprx
cd /aprx

Nu voeren we achtereenvolgens uit:

sudo ./configure
sudo ./make
sudo ./make install

Kontroleer of aprx is geïnstalleerd met ***sudo aprx –V***

```
pi@tnc-pi ~/aprx $ aprx -V
aprx: 2.8.1-W6KWF-10-g6f3a458
pi@tnc-pi ~/aprx $
```

Nu nog even de configuratie van aprx aanpassen. (/etc/aprx.conf)…

sudo nano /etc/aprx.conf

```
# Config file voor PA0ESH-11
mycall  PA0ESH-11
myloc lat XXXX.XXN lon XXXX.XXE

<aprsis>
        server          rotate.aprs2.net
        #server         aprs.pa4tw.nl
        passcode        XXXXXX   (XXXXX vervangen door passkode die bij call hoort)
                        # zie ook http://apps.magicbug.co.uk/passcode/
        filter  a/52.53083/06.34767/52.12633/07.226667
        heartbeat-timeout 1m
</aprsis>

<logging>
        pidfile /var/run/aprx.pid
        rflog /var/log/aprx/aprx-rf.log
        aprxlog /var/log/aprx/aprx.log
</logging>


<interface>
        ax25-device     PA0ESH-11
        tx-ok           true
        alias           RELAY,TRACE,WIDE
</interface>

en hier de rest van de config file.
```

Zoals je ziet, heb ik mycall, en de call van ax25 gelijk staan. Dat geeft voor mij de minste verwarring.
Op de Raspberry moet je ook nog even de log directory's aanmaken.
Dat doe je met :
sudo mkdir –p /var/log/aprx

Autostart APRX op Raspberry

Tenslotte moet het programma zelf starten bij het booten.
Maak een file aan in de directory **/etc/init.d** genaamd **aprx**
sudo nano /etc/init.d/aprx
Zet de volgende inhoud in deze file.

```
#!/bin/bash
# /etc/init.d/aprx

### BEGIN INIT INFO
# Provides:          aprx
# Required-Start:    $remote_fs $syslog
# Required-Stop:     $remote_fs $syslog
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: initscript to start aprx as daemon
# Description:       This service is used to manage aprx
### END INIT INFO


case "$1" in
    start)
        echo "Starting aprx 2.08"

        /usr/local/sbin/i2ckiss 1 16 vhf1 44.128.1.1
#       /usr/local/sbin/i2ckiss 1 32 vhf2 44.128.1.1   # tbv 2e TNC-PI
         sleep 5
        /sbin/aprx
        ;;
    stop)
        echo "Stopping aprx 2.08"
        killall i2ckiss
```

```
        killall  aprx
        ;;
    *)
        echo "Usage: /etc/init.d/aprx start|stop"
        exit 1
        ;;
esac

exit 0
```

Sla de file op en make 'm executable met

sudo chmod +x /etc/init.d/aprx

En als ie dus moet starten bij het booten, dan nog even

sudo update-rc.d /etc/init.d/aprx defaults

Voila, na een reboot:

```
pi@tnc-pi ~ $ ps aux|grep aprx
root      2168  0.0  0.4   2896  2036 ?        Ssl  16:34   0:00 /sbin/aprx
root      2169  0.0  0.2   1700  1196 ?        Ss   16:34   0:00 startpar -f -- aprx
pi        2213  0.0  0.4   3552  1836 pts/2    S+   16:37   0:00 grep --color=auto aprx
pi@tnc-pi ~ $
```

Als er nieuwe updates zijn van aprx, dan herhaal je het voorgaande proces, maar eerst vanuit de root directory van pi voer je uit : git pull. Daarmee komen de updates van de repository naar de Raspberry.

73'S
Erik, PA0ESH
www.pa0esh.nl